Software Requirements Specification

for

MapMyUNI

11 November 2024

Prepared by:

Specialization	SAP ID	Name
AIML	500105922	Devanshi Narula
AIML	500110860	Mukul Shivhare
AIML	500102030	Daksh Batra
AIML	500106774	Anvita Gupta

School Of Computer Science
UNIVERSITY OF PETROLEUM & ENERGY STUDIES,
DEHRADUN- 248007, Uttarakhand

TABLE OF CONTENTS

REVISION HISTORY

Date	Change	Reason for Changes	Mentor Signature
11-11-24	Initial Version	First Draft	

1. INTRODUCTION

1.1 Purpose of the Project:

The purpose of "Map My Uni" is to develop a navigation tool specifically designed for university campuses. This software provides the shortest possible route between any two locations within the university, leveraging Dijkstra's algorithm to calculate optimal paths. This will be beneficial for students, staff, and visitors who need efficient directions within the campus.

1.2 Target Beneficiary

The primary users are university students, faculty, staff, and visitors who may need guidance on the shortest or fastest route between campus facilities.

1.3 Project Scope

This software is designed to support users in navigating a complex university campus, enhancing convenience, and reducing time spent locating buildings or facilities. Key deliverables include a user-friendly interface, accurate maps, and pathfinding capabilities that take into account the latest campus layout.

1.4 References

- Dijkstra's Algorithm resources
- Existing university campus maps
- Relevant software engineering textbooks or documentation standards

2. PROJECT DESCRIPTION

2.1 REFERENCE ALGORITHM:

Dijkstra's algorithm is a classical algorithm used for finding the shortest path between nodes in a graph, which may represent, for example, road networks or network topologies. It is particularly useful for graphs with non-negative edge weights, such as distances between locations on a university campus. Here's a step-by-step explanation of how Dijkstra's algorithm works: 1. Initialization: • Start Node: Begin with the source node (the starting location). Set its distance to zero because the distance from the source to itself is zero. • Distance Table: Initialize a distance table where each node's distance is set to infinity (∞), except for the source node, which is set to zero. • Priority Queue: Use a priority queue (or a min-heap) to keep track of nodes to be processed. Initialize this queue with the source node.

2.2 Data/Data Structure:

To implement the project of calculating the minimum distance between two places on a university campus, appropriate data and data structures must be utilized to represent the campus layout and facilitate efficient computation using Dijkstra's algorithm. Below is a

description of the sample data and the native data structures suitable for the project. Considering the UPES BIDHOLI campus with the following locations and pathways

2.3 Swot Analysis:

A SWOT analysis evaluates the Strengths, Weaknesses, Opportunities, and Threats related to the development and implementation of a campus navigation tool based on Dijkstra's algorithm. This analysis compares the proposed tool with existing solutions to highlight its advantages and identify areas for improvement.

2.3.1 Strengths:

- Accurate Shortest Path Calculation: o Advantage: Dijkstra's algorithm is well-suited for finding the shortest path in graphs with non-negative weights, ensuring accurate distance calculations. 8 o Comparison: Unlike heuristic-based methods or simple Euclidean distance approaches, Dijkstra's algorithm guarantees optimal solutions for shortest path problems.

- Efficiency: Advantage: With a time complexity of O((V+E)logV), the algorithm is efficient for handling graphs with a moderate number of nodes and edges. o Comparison: Compared to brute-force methods, which have exponential complexity, Dijkstra's algorithm provides a significant performance advantage in real-time applications.

- User-Friendly Interface: o Advantage: The program can feature an intuitive interface for easy input and navigation, making it accessible to users with varying levels of technical expertise. o Comparison: Existing solutions such as paper maps or manual directions lack interactive and dynamic features, making them less user-friendly.

- Real-Time Updates: o Advantage: The program can be updated to reflect changes in the campus layout or new pathways, ensuring the information remains current. o Comparison: Static maps or traditional navigation aids do not offer real-time updates, potentially leading to outdated information.

2.3.2 Weaknesses:

- Setup Complexity:
    - Disadvantage: Creating and maintaining an accurate graph of the campus with all pathways and distances is a valuable investment of time that will provide lasting benefits.
    - Comparison: Some existing campus maps or GPS-based systems may already have data available, allowing for a quicker setup while still offering valuable insights.

- Scalability:
    - Disadvantage: As the campus grows or becomes more complex, the graph may become large and require more computational resources. o Comparison: Simple solutions or less detailed maps may perform better in smaller or less complex environments.

- Dependency on Accurate Data:
    - Disadvantage: The accuracy of distance calculations depends on the precision of the input data. Errors in data collection can lead to incorrect results.
    - Comparison: Some existing systems might use more robust data sources or incorporate error-checking mechanisms.

- User Dependency on Technology: Disadvantage: The tool requires users to have access to a device with the program installed, which may not always be feasible. Comparison: Traditional navigation methods like printed maps do not rely on technology, making them universally accessible.

2.3.3 Opportunities:

- Integration with Other Systems:
    - Opportunity: The tool can be integrated with other campus systems such as event schedules, building directories, and emergency alerts.
    - Comparison: Unlike standalone systems, integrated solutions offer enhanced functionality and user experience.
- Enhanced Features:
    - Opportunity: Future versions could include features such as voice navigation, accessibility options for disabled users, and integration with mobile apps.
    - Comparison: Existing solutions may lack these advanced features, giving the tool a competitive edge.
- Improved User Engagement:
    - Opportunity: Interactive maps can enhance user engagement and satisfaction, leading to better campus navigation.
    - Comparison: Less interactive systems might not engage users as effectively.
- Customization for Different Campuses:
    - Opportunity: The tool can be customized and adapted for different university campuses, making it a versatile solution.
    - Comparison: Generic navigation systems may not provide the level of customization needed for specific campus layouts

2.3.3 Threats:

- Technological Challenges:
    - Threat: Technical issues such as software bugs, data corruption, or device compatibility problems could affect the reliability of the tool.
    - Comparison: Traditional methods are less prone to technical failures but may lack advanced functionality.
- User Adoption:
    - Threat: Users may be resistant to adopting new technology or may prefer existing methods, limiting the tool's effectiveness.
    - Comparison: Established methods might have an ingrained user base, making it challenging to transition to a new tool.
- Data Security:
    - Threat: Handling user data and campus layout information poses a risk of data breaches or privacy concerns.
    - Comparison: Systems that are not connected to external networks may be less vulnerable to data security threats.
- Cost of Development and Maintenance:
    - Threat: The costs associated with developing, deploying, and maintaining the tool could be significant.
    - Comparison: Existing systems with lower initial investment might be more cost-effective, though they may lack advanced features.

2.4 Project Features

- Interactive Campus Map:
    - Users can view a visual representation of the UPES BIDHOLI campus with labelled locations like buildings, food courts, hostels, etc.
    - Map markers or icons represent major landmarks, allowing users to click or tap for more information.
- Location Search:
    - Users can search for specific campus locations by name (e.g., "Library," "MAC").
    - The map highlights the searched location, providing relevant details.
- Shortest Path Calculation:
    - Users can select a starting location and a destination.
    - The system calculates and displays the shortest path between these two points using Dijkstra's algorithm.
    - This path can be visually highlighted on the map, with estimated travel distance and time.
- Pathway Information:
    - Users can view information about pathways, including the distance between different nodes.
    - Details of alternative routes are provided when available.
- User-Friendly Interface:
    - A clean, intuitive UI design for easy navigation through different map features.

    - Simple controls and search functionality for ease of use.
- Location and Path Updates:
    - Option for admins to update locations, add new paths, or modify existing ones if there are changes on the campus (like new buildings or blocked paths).
- Campus Overview Section:
    - Basic campus information for new students and visitors, with key landmarks and amenities highlighted.

2.5 User Classes and Characteristics

- Students:
    - Description: Primary users who need help navigating the campus, especially first-year or transfer students who may be unfamiliar with the layout.
    - Characteristics: Likely to use the app frequently for finding classrooms, libraries, cafeterias, and other essential locations. Expect a user-friendly interface with a quick search and easy pathfinding options.
- Faculty and Staff:
    - Description: Includes professors, administrative staff, and other campus personnel who may need to find specific buildings or locations for meetings or events.
    - Characteristics: Less frequent but specific use cases, such as finding alternate paths in case of road closures or meeting points. Expect an easy-to-use map with accurate campus information.
- Visitors:
    - Description: Parents, event attendees, or guest lecturers who visit the campus occasionally.
    - Characteristics: Unfamiliar with the campus layout and likely to need the shortest path to a destination from entry points like the main gate. Expect straightforward navigation and intuitive instructions.

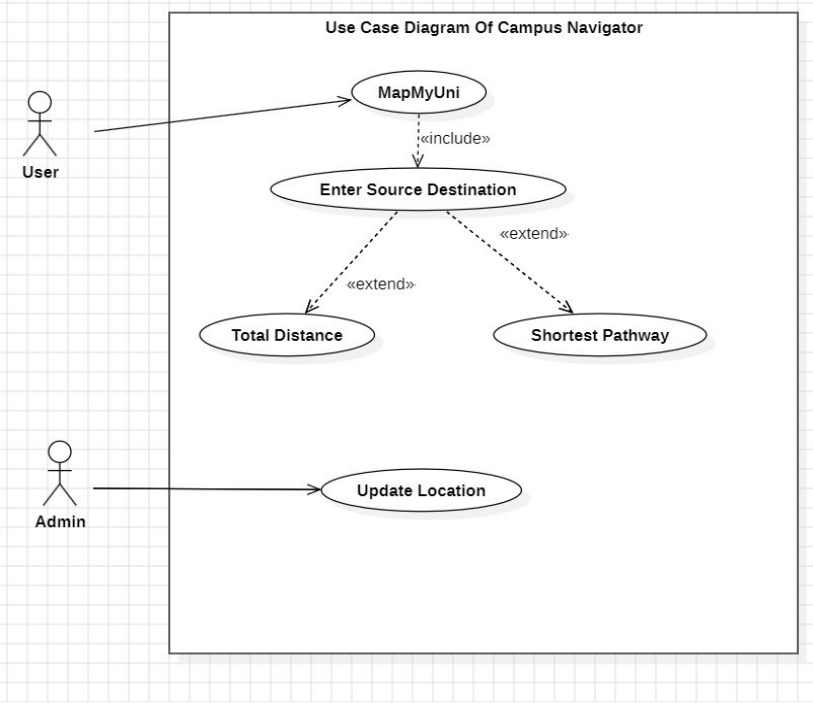
2.6 Design and Implementation Constraints

- Platform Constraints:
    - The project is intended for specific platforms, which may limit certain design aspects or functionalities depending on device compatibility and screen size.
- Algorithm Complexity:
    - Dijkstra's algorithm, while effective for shortest path calculations, has time complexity constraints, especially with a large number of nodes and edges. The design must be optimized to handle potential scalability issues if more locations are added.
- Resource Limitations:
    - Limited availability of campus data may restrict the map’s accuracy and the granularity of location details.

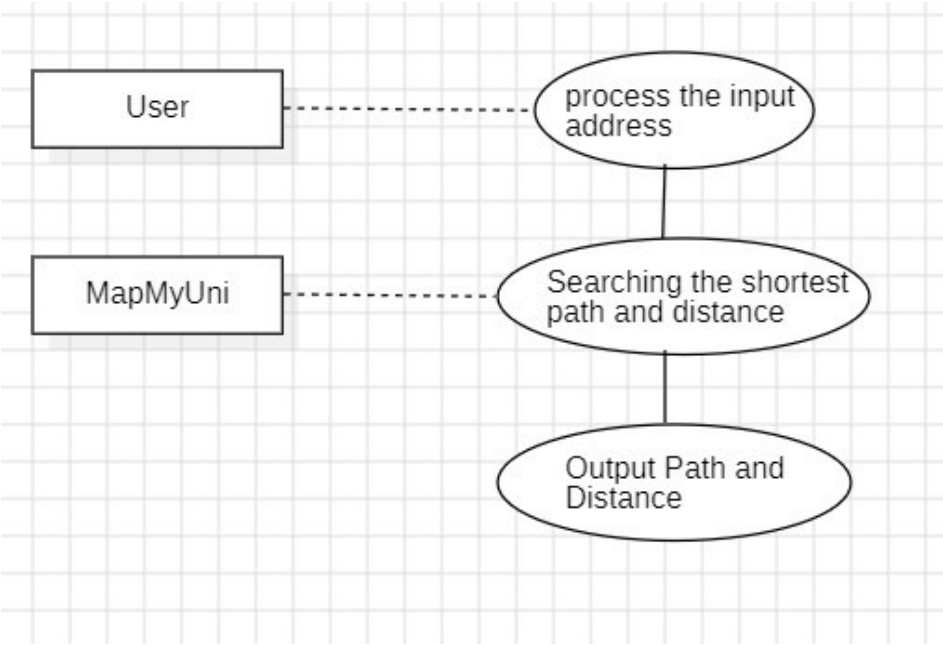
- Usability Constraints:
    - The app must be designed to provide an intuitive user experience that minimizes user actions (clicks or taps) to find destinations and view paths, catering to various users' technical proficiency levels.

2.7 Diagrams

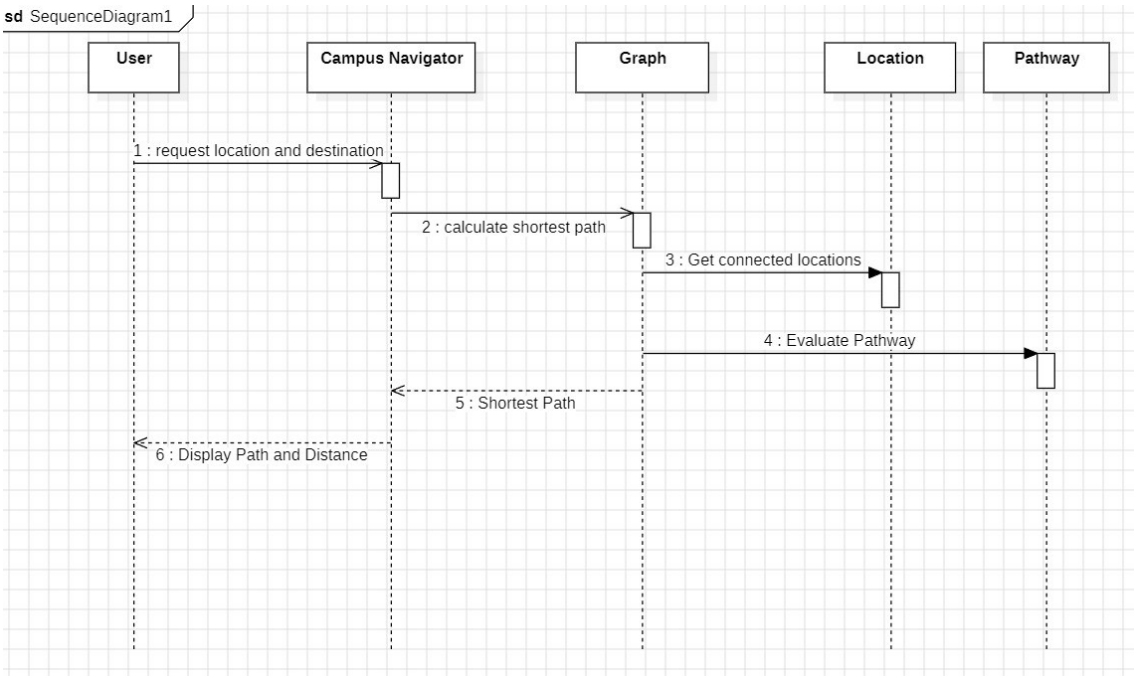
2.7.1 Use-Case Diagram



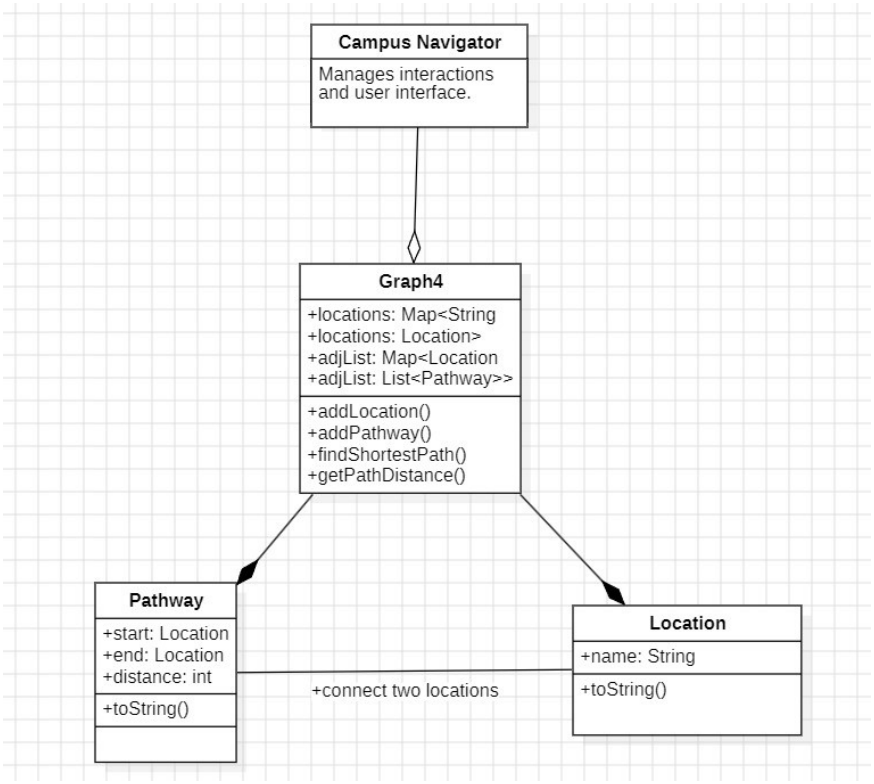
2.7.2 Data Flow Diagram



2.7.3 Sequential Diagram



2.7.4 Class Diagram



2.8 Assumptions and Dependencies

- Assumptions:
    - The campus layout and locations remain stable with minimal changes over time.
    - Users have basic internet access if required for online features.
    - Users are familiar with general map usage and navigation features.
    - The data provided for campus locations and distances is accurate and up-to-date.
- Dependencies:
    - Reliable data for campus paths and distances to ensure accurate navigation.
    - Availability of chosen libraries or frameworks for UI and algorithm implementation.
    - Admin support for updating map data in case of campus modifications.

3. SYSTEM REQUIREMENTS

3.1 User Interface

- Map Display: The main screen displays the campus map with clickable or tappable markers for key locations.
- Search Functionality: A search bar allows users to find specific locations by name, highlighting the result on the map.
- Path Visualization: On selecting a starting point and destination, the shortest path is visually highlighted on the map.

3.2 Software Interface

- Programming Languages: Uses JavaScript for web-based implementations or Swift/Kotlin for mobile apps.
- Frameworks/Libraries: Integration with mapping libraries (e.g., Leaflet, Google Maps API) for map functionalities, and a graph library if needed for pathfinding.
- Operating System Compatibility: Compatible with major OS platforms (Windows, macOS, iOS, Android) if the app is cross-platform.

3.3 Database Interface

- Database Type: Use of a relational database (e.g., MySQL) to store data for campus locations, distances, and paths.
- Data Access: Database connections enable real-time retrieval and updates of location and pathway data.
- Admin Access: Secure database access for admins to add, update, or remove location data.

3.4 Protocols

- Data Transmission: HTTP/HTTPS protocols for secure data exchange between the server and client applications.
- API Communication: If external APIs (e.g., Google Maps API) are used, secure API calls ensure data is exchanged reliably.
- Authentication: Admin access requires secure login protocols to prevent unauthorized changes to the campus map data.

4. NON-FUNCTIONAL REQUIREMENTS

4.1 Performance Requirements

- Response Time: The system should respond to user inputs (search, map interaction) within 2-3 seconds.
- Scalability: The system should be able to handle a growing number of locations or pathways without significant performance degradation.
- Load Handling: The system should efficiently handle up to 100 concurrent users, ensuring no noticeable lag or downtime.

4.2 Security Requirements

- Data Security: Any user data (if collected) should be encrypted during transmission and stored securely in compliance with privacy standards (e.g., GDPR).
- Authentication: Admins must authenticate using a secure login method to access sensitive features like data editing.
- Access Control: User access is restricted based on roles (e.g., admins vs. regular users) to prevent unauthorized changes.

4.3 Software Quality Attributes

- Usability: The system should have a user-friendly interface that requires minimal training or explanation. Navigation features should be intuitive for first-time users.
- Reliability: The system should have a 99% uptime, ensuring continuous access for users. Errors and bugs should be minimized and fixed promptly.
- Maintainability: The system should be easy to update and maintain, allowing for quick additions or modifications to locations and pathways without major downtime.
- Portability: The system should be portable across different platforms (e.g., web, Android, iOS) without requiring major changes to the core functionality.

Appendix A: Glossary

- MapMyUNI: A navigation tool designed specifically for university campuses to provide directions within campus boundaries.
- Dijkstra's Algorithm: A shortest-path algorithm used in MapMyUNI to find the quickest route between two locations within the campus.
- User Interface (UI): The visual part of the software that users interact with, including map display, search options, and navigation.
- Pathfinding: The process of calculating the shortest route from one location to another.
- Node: A point on the map that represents a location or intersection within the campus.
- Edge: A connection between two nodes on the map, representing a pathway or road.
- Priority Queue: A data structure used to manage the nodes in Dijkstra's algorithm for efficient pathfinding.
- Database: A structured collection of data, used here to store information about campus locations and pathways.
- Admin: A user role with elevated privileges for updating campus data, paths, and location information within the software.
- HTTPS: A protocol for secure communication over the internet, ensuring that user data is protected during transmission.

Appendix B: Analysis Model

The analysis model for MapMyUNI outlines the processes and interactions involved in campus navigation. The core components include:

- Use-Case Diagram:
    - Displays various user interactions, such as students searching for locations, faculty finding meeting points, and admins updating paths.
    - Main use cases include Location Search, Shortest Path Calculation, and Admin Data Update.
- Data Flow Diagram (DFD):
    - Level 0 DFD shows the input (location search) and output (shortest path) processes for a user.
    - Level 1 DFD expands to show data retrieval from the database, path calculation using Dijkstra's algorithm, and map update processes.
- Class Diagram:
    - The system has classes for User, Admin, Location, Path, and Map.
    - User interacts with Location and Path classes to calculate routes.
    - Admin has privileges to interact with Location and Path for updates and changes.
- Sequence Diagram:
    - Illustrates the sequence of interactions for common actions, like a user searching for a location or an admin updating a path.
    - Shows request and response flows between the User Interface, Database, and Pathfinding Algorithm.

Appendix C: Issue List

Problem Definition and Scope:

- Defined the problem of complex campus navigation and the need for an efficient tool to calculate the shortest paths between key campus locations.
- Established project objectives, including simplifying campus navigation, reducing travel time, and enhancing campus accessibility.

2. Requirements Gathering:

- Created a Software Requirements Specification (SRS) document in line with standard guidelines.
- Documented functional requirements for core features:
- Input of start and destination locations.
- Calculation of the shortest path based on real-time data.
- Displaying the calculated route and distance.
- Identified non-functional requirements, including:
- Performance requirements for real-time calculations.
- Security and data protection measures.

3. System Design:

- Selected Dijkstra's algorithm as the core pathfinding method due to its efficiency with graphs featuring non-negative weights.
- Designed the data structure:
- Graph class, using an adjacency list to represent locations (nodes) and pathways (edges).
- Priority Queue for efficiently retrieving nodes with minimum distances during pathfinding.
- Distance Table for tracking shortest distances from the source node.
- Conducted a SWOT Analysis to assess strengths, weaknesses, opportunities, and potential threats for the project.

4. Diagram Creation:

- Developed system design diagrams as per the SRS requirements:
- Use Case Diagram for illustrating user interactions with the system.
- Class Diagram depicting the main classes (Graph, Location, Pathway, and MapMyUni) and their relationships.
- Activity Diagram demonstrating the steps involved in calculating and displaying the shortest path.
- Sequence Diagram for detailing the sequence of interactions between components.
- Data Flow Diagram (DFD) visualizing data inputs, processing, and outputs.

5. Code Implementation:

- Developed Java code for key functionalities:
- Created Location and Pathway classes to represent campus points and paths.
- Implemented the Graph class to manage locations and pathways, with methods for:
    - Adding locations and pathways.
    - Finding the shortest path using Dijkstra's algorithm.
    - Calculating and displaying path distances.